City, University of London Institutional Repository

This is the published version of the paper.

This version of the publication may differ from the final published version.

VisUnit: Literate Visualisation Studies Assembled from Reusable Test-Suites

Radu Jianu
City St George's, University of London
London, United Kingdom
radu.jianu@city.ac.uk

Aidan Slingsby
City St George's, University of London
London, United Kingdom
a.slingsby@city.ac.uk

Dany Laksono
City St George's, University of London
London, United Kingdom
dany.laksono@city.ac.uk

Mershack Okoe
Florida International University
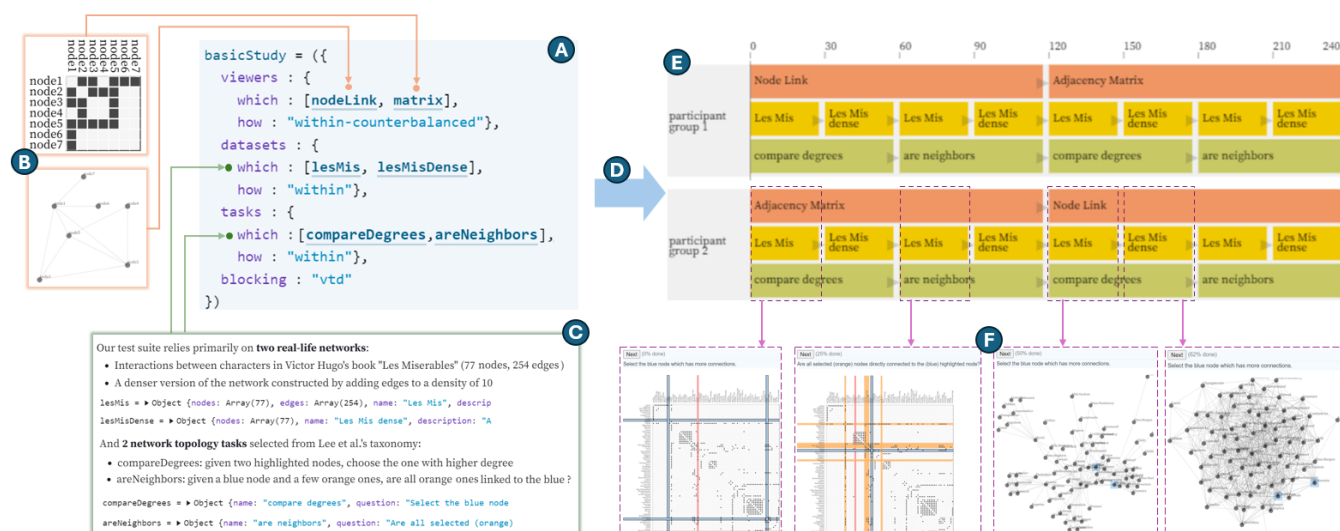Miami, Florida, USA
meshhome16@gmail.com

Figure 1: (A) Visualisation user studies can be assembled using a declarative specification from (B) visualisations and (C) separately developed benchmark data and tasks used to evaluate them; the latter can be organised into 'literate' test-suites that combine test data with narratives explaining their use and justifying their choice. (D) VisUnit parses declarative studies into (E) study sequences and delivers (F) visual stimuli to participants by combining visualisations, datasets, and tasks dynamically.

Abstract

We make four contributions to lower the overhead of conducting visualisation user studies and promote the reuse and extension of their materials. **(i)** A declarative Javascript specification lets experimenters describe how studies are assembled from tested visualisations, datasets, tasks and chosen evaluation strategies. **(ii)** A VisUnit library translates these into sequences of visual stimuli and delivers them to participants. We move away from monolithic evaluation stimuli typical of previous work and construct studies around three ingredients – visual encodings, datasets, and tasks – that can be developed independently and recombined flexibly. **(iii)** This paves the way for developing benchmark *data+tasks test-suites* as independent, reusable resources to support multiple studies. **(iv)** Structuring user studies as "literate" visualisation notebooks brings together in the open all ingredients necessary for replication and scrutiny: formal design specification; underlying materials; participant-facing views; and narratives justifying design and supporting reuse.

CCS Concepts

• **Human-centered computing** → *Visualization design and evaluation methods*; **Empirical studies in visualization**; Visualization systems and tools.

Keywords

visualisation evaluation, user studies

1 Introduction

User studies are commonly employed to formally evaluate the effectiveness of visualisation systems and unravel how our perceptual and cognitive systems interact with visual data representations [24, 33]. Over the last decade more have been published each year [44] and many are influential on visualisation research. However, empirical evaluation work trails behind technique development and stifles the evolution of a cohesive understanding of how visual tools should be designed [21, 30, 44]. This is attributed to studies being challenging to conduct, time consuming to carry out, and their reliance on a broad range of activities and expertise. There is also a lack of published study materials, data, and detailed analyses that future researchers can build on [31]. We introduce **four contributions** to alleviate this:

- **(i) a declarative grammar for specifying visualisation studies** from constituent components (tested visual encodings, datasets and tasks) and evaluation strategies (randomisation and blocking approaches, training) and
- **(ii) VisUnit, a library** that automatically assembles theses into web-studies that can be readily delivered to participants.

These build on earlier ideas from EvalBench [2], GraphUnit [41], Touchstone2 [17], EvalViz [37] and, more recently, reVisIt [15]. Our declarative language and VisUnit share benefits already evidenced: they lower the overhead of assembling and deploying user studies; create a transparent link between a user study's formal design and its delivery; and support the exploration of alternative study designs by letting experimenters reshape studies with minimal effort. VisUnit innovates by moving away from monolithic visual stimuli, instead allowing them to be declared in terms of constituent components – encoding (and interaction), datasets, and tasks – and combining these into studies dynamically, based on chosen evaluation strategies. This decomposition fits models of visualisation designs [40, 46, 53] and inspires a new reuse opportunity:

- **(iii) data+task test-suites as research contributions and resources** in their own right.

Visualisation studies most often source their own datasets and tasks to evaluate with. This is non-trivial and time-consuming [44], as the materials need to be ecologically valid yet constrained enough to support evaluation, minimise confounding factors, and afford meaningful participant responses to data questions. We advocate that researchers pursue *data+task test-suites* – concrete datasets and questions about them – as contributions in their own right. Comprehensive test-suites for particular types of data (e.g., multidimensional, network, spatio-temporal) could, once developed, underlie a body of cohesive evaluative research on how best to visualise those data. Pursuing them independently, rather than as by-products of specific studies, would focus effort, expertise and scrutiny on assembling test-suites meaningful, comprehensive, and rigorously built. Moreover, they connect with efforts to develop *visualisation task taxonomies* by providing for them concrete and representative exemplars that visualisations can readily test against.

- **(iv) literate user studies** that showcase and explicitly discuss the impact of selected and competing experimental design choices on study delivery and which bring together and in the open all ingredients necessary for replication and public scrutiny.

When reporting on user studies, authors typically showcase them as delivered to participants and motivate the chosen experimental design over *imagined* alternatives. Instead, specifying studies declaratively lets alternative designs be explored and experienced directly to strengthen the case for the chosen option. This is possible because design adjustments (e.g. evaluating a factor between participants rather than within) can be translated automatically into revised study deliveries. We show that visualisation notebook environments such as *Observable* [1] can bring together all ingredients necessary for a study's replication and public scrutiny: its formal design specification, underlying study materials (visual encodings, data, and task specifications), alternative choices, how these present to participants, and narratives justifying choices and supporting reuse.

Our approach borrows nomenclature and ideas from *Unit Testing* in software development. Our *data+task test-suites* are 'live', testable instantiations of abstract task taxonomies in the same way that test-suites are developed for software projects as 'live' representations of static project-requirement documents. We let visualisations connect and evaluate against *data+task test-suites* automatically using VisUnit in the same way that software test-suites are managed by Unit testing libraries (see Section 2).

2 Related work

2.1 Visualisation user studies and their challenges

Quantitative user studies in data visualisation typically involve showing people (participants) visualisations of data and measuring their performance (accuracy, speed) and preference as they attempt tasks that probe their understanding of the data. They play a central role in establishing the effectiveness of data visualisation designs [24, 33, 39].

Conducting user studies is challenging, time consuming, and requires diverse expertise [8, 17, 31, 32, 44]. They involve a broad range of activities such as scoping the research, sourcing or implementing visualisations to evaluate, coming up with tasks and datasets that are representative of how visualisations are used in practice yet can support the generation of quantifiable insights, designing study procedures that reduce confounding factors, building infrastructure to run the study and collect results, recruiting and managing participants, and analysing results. A rigorously conducted, documented, and disseminated study is multi-month-long process that requires diverse expertise [44].

Studies are often difficult to replicate or extend. Following a survey of visualisation studies on perception, Quadri *et al.* conclude that one of the biggest challenges facing such studies is their limited scope and reproduceabilty. They attribute this to the difficulties in sourcing study materials and accessing diverse participants [44]. In their review and positions on the state of replication in visualisation,

---

[1] www.Observablehq.com

Kosara and Haroz advocate for replication as a viable visualisation contribution. They deem access to original data, code as well as documented methods and analyses as crucial to support this [21, 30].

Our work tackles these challenges holistically. *VisUnit literate studies* bring together in the open all ingredients needed for public scrutiny and seamless reuse and extension. *VisUnit test suites* elevate the design of test data and tasks to contributions in their own right, ready to support evaluation efforts pursued independently (Section 4).

2.2 Supporting the creation and replication of user studies

Prescriptive guidelines for conducting user studies include best practices for designing materials and study flows [4, 31], for ensuring that studies can be replicated and extended [21, 30], and for moving from in-lab to online studies [9, 23, 32]. VisUnit supports the adoption of prescriptive methods by: (1) lowering the overhead of exploring alternative study designs through the use of declarative specifications, and (2) by asking researchers to justify design choices in-situ with the user study and its materials (Sec.4,3.2,5). The first contribution is not entirely new. The authors of Touchstone2 [17] remarked that the ease of manipulating study designs in declarative forms supports the exploration of alternatives. However, our fourth contribution is to make such exploration part of explicit, open design narratives and place these at the forefront of *literate study* dissemination.

Efforts to lower the overhead of creating, deploying, and processing the results of user studies also exist. Most relevant are frameworks that move away from stimuli-sequences that are crafted manually for different participant groups (treatments), to assembling sequences flexibly from study-components and rules described in *declarative study specifications*, and deliver them with little overhead to participants.

EvalBench [2] was to our knowledge the first to use elements of these innovations. It relied on XML definitions of participant tasks and a Java architecture that could transform these into visual stimuli. Our own *GraphUnit* [41] explored the assembly of network visualisation studies from XML study-design specifications and a selection of reusable network datasets and parametric tasks informed by literature. Subsequently, *Touchstone2* used a declarative language to describe randomisation strategies for use in Human Computer Interaction experiments and remarked how it lets researchers explore alternative designs with ease. Recently, *reVisIt* adopted similar ideas but innovated with a more powerful design-specification form in modern JSON to support the assembly of studies from a wide range of web-stimuli [15]. *EvalBench*, *Graph-* and *VisUnit*, and *reVisIt* all introduce some form of parameterised task definitions. Section 3.2 describes how this works in VisUnit. Overall, these innovations significantly streamline the creation of user studies.

Other research efforts to support the creation of user studies are less closely related to ours and rely predominantly on visual stimuli assembled *a priori* into static images [12, 13, 18, 37, 49, 50]. Finally, as already argued by Ding et al [15] commercial surveying platforms such as *Qualtrics* are not sufficiently flexible to support the needs of complex visualisation evaluation.

Our contribution to this space is two-fold. First, our declarative form shares the benefits of those proposed previously: they capture study designs formally and support the exploration of alternative designs. However, we add one more step towards fully modular user-study designs. VisUnit's decomposition of visual stimuli into their constituent parts – visualisations, datasets, and tasks – inspired us to move away from monolithic studies and separate out *test-suites of datasets and tasks* as reusable, extensible research artefacts and resources. Second, we show how declarative study-specifications can underlie a new form of *literate design and dissemination of studies and their materials*.

2.3 Synergies with visualisation task taxonomies

Our *data+task test-suites* complement work on *visualisation task taxonomies*. These refer to systematic classifications of tasks or objectives users pursue while analyzing and interpreting data visually. They help designers create visualizations that align with user needs and inform their evaluation. Some task taxonomies are generic. They encompass broad activities like exploration, comparison, and pattern recognition applicable across different visualisation types [3, 11, 52, 56]. Others are tailored to specific kinds of data, such as network [1, 26, 34], geospatial [5, 45], or temporal data[38, 47] to capture their particularities.

Many user studies rely on such taxonomies to derive tasks to evaluate against [44]. However, **the journey from abstract task taxonomies to concrete task presentations** that are instantiated with data and can be deployed in a user study **is arduous** and can account for the most significant part of a study's design [44]. First, while most taxonomies exemplify task categories, they do not do so exhaustively. It is often difficult to imagine the ways in which an abstract task can present in practice, especially when considering the nuances of real-datasets. Second, the exact way in which a task presents to participants needs to be designed carefully to reduce the impact of confounding factors. Finally, picking datasets and instantiating tasks in them, i.e. picking which data facets a participant is queried about, is often non-trivial since tasks rely on answers being possible and meaningful in the dataset.

VisUnit lets such activities be carved out as research in its own right, distinguishable from other evaluation efforts. Essentially, *data+task test-suites* can act as instantiations of visualisation task taxonomies into datasets, that can be disseminated as stand-alone evaluation resources for visualisations to test against. Moreover, test-suites can go beyond supporting evaluation to serve as concrete exemplars of how abstract task-taxonomies translate to practice.

2.4 Parallels to Unit Testing

Our work was further inspired by *Unit Testing* concepts in software development [27]. This involves testing 'units of code' (methods, classes, packages) to ensure they meet their functional requirements. Concretely, formal tests are designed for each unit of a software project to verify that the unit's outputs for some given inputs match expectations. Individual tests are then collected in a *test-suite*, which a project's code base can be tested against repeatedly, as it is changes and evolves. Developing reliable and exhaustive tests is non-trivial but once a *test-suite* assembled, repeated testing can be done at no

added cost, with the help of language-specific unit testing libraries. Since tests are typically created to meet a project's documented requirements, *test-suites* are seen as "live" representations of project requirements, ones that software can be readily evaluated against. In fact, in test-driven development, unit tests are written before the software is even developed as a means of capturing how the software should behave in practice.

We transfer two fundamental ideas to visualisation evaluation. First is the ability to capture the data-reading requirements of visualisations – what they need to answer about the data – in concrete, "live" *test-suites* that visualisations can test against. Second is the ability to connect visualisations to such test-suites and test against them with minimal overhead using "unit" libraries. We acknowledge the parallelism does not extend completely. User-studies remain reliant on participant engagement and are thus unlikely to be used as freely and often as unit-testing. Nevertheless, we see value in borrowing those aspects that *do translate* and can benefit visualisation evaluation.

Finally, we note that unit testing and visualisation user studies bear resemblance in their focus on evaluating relatively low-level functionality robustly. They both occupy distinct roles among a wider array of testing options and typically are not meant to substitute methods that evaluate systems more holistically (e.g., system or usability testing in software development; insight-based and longitudinal engagement with users).

2.5 Building on literate visualisation and Observable notebooks

Literate visualisation [54] and visualisation notebooks more generally [29], rely on combining the narrative clarity of written language with the expressive power of visualizations as a means to discuss and justify visualisation designs or to create compelling data stories. Observable notebooks are a concrete technological embodiment of this paradigm. They allow users to combine data, code, visualisations, and explanatory text in a single document to integrate visual data analysis and storytelling. By extension, we call for "literate" user studies and visualisation *data+tasks test-suites* that combine explorable, ready-to-use study materials and design specifications with narratives that justify their choice and explain their use (Section 4). While the principles around our approach are technology-agnostic, we exemplify how it can work within the Observable environment and show that Observable's support for notebook development and extension – versioning, forking, tracing, and permission management – can be readily harnessed to support the creation of user studies that are easier to extend and reproduce (Section 5).

2.6 Other efforts to evaluate data visualisations

Finally, we note efforts to reduce the need for user-driven evaluation altogether. Wu *et al.* [55] review the use of AI to evaluate visualisations based on design guidelines or models trained on available empirical data and conclude that studies are still indispensable to the evaluation of complex and novel visual encodings. Heuristic approaches [51] rely on expert designers' assessment of a visualisation's expected effectiveness. Data saliency models attempt to capture the working of people's visual perception and predict the effectiveness of visual encodings [35]. More relevant to our work are methods that rely on inspection of visualisation code or design such as algebraic visualisation evaluation and linting. They target similarly low level data-reading tasks as quantitative user studies do and also draw on parallels to software development. Visualisation linting [36] relies on close code/visualisation inspection and recommendations for improvement while algebraic evaluation [28] is in some ways similar to formal program verification [16] in that it attempts to use rules to determine the effectiveness of a visualisation/program without actually running it. We argue that linting and algebraic visualisation evaluation, while useful, cannot substitute the need for actual user evaluation data, especially in more complex usage scenarios; e.g. those involving interaction.

3 Declarative visualisation user studies

Stimuli typical of visualisation studies can be thought of as a **combination of data, visual encoding, and tasks**. We show how we can define these separately and **assemble them flexibly into different combinations** (Section 3.1). This paves the way towards describing user studies declaratively based on evaluated visualisations, datasets, and tasks and associated testing strategies (e.g. *between* or *within* participant testing; whether tasks are timed and trained). We introduce a **concrete declarative specification** that can capture a broad range of visualisation studies and a **Javascript VisUnit library** that can interpret the specifications, translate them into studies, and deliver them to participants (Section 3.2). We **validate** these contributions in Section 3.4. We first fully replicate two previously published studies [20, 43]. We then sample 49 representative visualisation user studies from TVCG and discuss the degree to which VisUnit can replicate them.

3.1 Assembling visual stimuli from constituent components

Visual stimuli are the backbone of visualisation user studies. Most can be seen as combinations of a *(a)* visual encoding of some *(b)* underlying data and *(c)* elements pertaining to a task that participants are asked to do. The latter may include task instructions, annotations on the primary visualisation (e.g., highlighted marks that task instructions reference), and mechanisms for answering. Example stimuli are shown in Fig. 2. This delineation between visualisation, data, and tasks aligns with conceptual models of data visualisation design [40, 53] and efforts to develop task taxonomies [3, 11, 34, 45, 52, 56]. With a few exceptions, most visualisation studies report on their stimuli and materials along these three facets.

Studies typically expose participants to multiple 'repeats' (or 'trials') which are visual tasks of similar form but with different content and answers. For example, Ghoniem *et al.* generate a new random network for each repeat of the same task [19]. Okoe *et al.*[42] asks participants to find paths between three distinct pairs of nodes selected from the same network. Tory *et al.* [48] structure multiple repeats of a spatialisation task around sub-regions within a single large dataset.

VisUnit task can be defined using a **parametric form** that: (i) can be defined independently from data the task may refer to and visualisations it may be pertain to; and (ii) supports the creation of

Figure 2: Examplar stimuli used in visualisation studies: (a) Okoe *et al.*'s stimuli emerge as combinations of different networks shown with different encodings [42]. (b) Beecham *et al.* construct line-ups of synthethic colorpleth maps that are auto-correlated [7]. Jianu *et al.* compare four different ways of encoding group information onto a given node-link diagram [25].

multiple repeats for the same task. Its design rests on insights from work on task taxonomies, recognising that concrete tasks emerge by instantiating generic ones (Amar et al.'s *pro-forma tasks* [3]; Roth's *operators* [45]) into concrete data elements within a dataset (Brehmer and Munzner's *inputs* and *outputs*; Roth's *operands*). Fig. 3 (upper-right) exemplifies the parametric form of a network task asking participants to determine which of two highlighted nodes has a larger degree (compare-degree). It includes its question to participants and two elements relating to its parameterisation:

- A set of **task inputs** (or parameters). Giving values to those inputs instantiates the task into different repeats. For example, the prototypical task *Find and select the node labeled $0* (find-node) would be instantiated by replacing $0 with different concrete node labels; the task *Determine which of the two highlighted nodes has more connections and select it* (compare-degree) would be instantiated by highlighting different pairs of nodes in the visualisations (see Fig. 3).
- A set of **task outputs** or responses expected from participants. These may be inputs into graphical widgets, keyboard interactions, or interactive changes to visualisation state. For example, the tasks exemplified above require participants to select their answers in the visualisation so their *outputs* are selected network nodes. Multiple output are possible such as quantitative answers accompanied by qualitative reflections (see Fig. 8).

To create concrete visual stimuli, parametric tasks are then **instantiated** into datasets, by picking concrete elements within those datasets as values for a task's inputs. Fig. 3 (bottom right) shows four repeats of the compare-degree task instantiated with inputs chosen from two different datasets; these were attached to the parametric task description. Concrete task stimuli can then emerge from the combination of independent visual encodings (viewers), datasets, and tasks and their instances. Fig. 3 exemplifies this: four distinct stimuli were assembled from one viewer (a node-link diagram) encoding two datasets and highlighting two task instances (pairs of nodes for a compare-degree task) for each of those datasets.

To achieve this in practice, VisUnit relies on viewers, datasets and tasks to implement a set of **interfacing conventions**. For example, all viewers that wish to interface with VisUnit need to

implement a loadData method (among a few others) which VisUnit would call when the study design asks for a new dataset to be shown. For a viewer to be able to set-up a task, it needs to implement 'setters' for its inputs and 'getters' for its outputs. Concretely, the viewer that produced the stimuli in Fig. 3 had to implement setHighlightedNodes() and getSelectedNodes() in order to set up the *compare-degree* task exemplified. The benefit of Javascript is that adding such methods to already implemented resources is relatively effortless. VisUnit relies on calls to these capabilities to piece stimuli together.

Finally, we find it useful to let tasks incorporate computation to determine whether answers provided by a participant (the *output*) are correct. As exemplified in Fig.3, the compare-degree task described above can verify that the node selected by a participant is indeed the more connected one and return an appropriate accuracy measure.

3.2 Declarative study specifications

As discussed in Section 2, previous research found benefits to designing studies using declarative specifications. Our declarative form integrates elements from such work. For example, parametric tasks in the context of declarative study designs were introduced in EvalBench [2] and adopted inGraphUnit [41] and reVisIt [15]. Centering study design around study blocks that are arranged in accordance to specified experimental designs and randomisation strategies was used in GraphUnit, Touchstone2 [17] and reVisIt. Additional similarities, in particular to reVisIt are discussed in Section 6. The main innovation in VisUnit and its declarative form stems from organising studies around viewers (visual encoding with interaction), datasets, and tasks and the benefits that this affords.

VisUnit study specifications start from factorial designs centered around the three factors described above: **viewers**, **datasets**, and **tasks**. Section **??** shows that variations of factorial designs form a large part of visualisation studies. Fig. 4 (left) illustrates a VisUnit declarative design in its simplest form. It captures a **2 viewer × 2 dataset × 2 task** design with all factors evaluated **within** participants and viewer counterbalancing. The blocking hierarchy is viewer→data→tasks (*blocking : "vtd"*). This means that participants will complete all activities for the first viewer before moving to the next. For each viewer, they will complete each task for all datasets
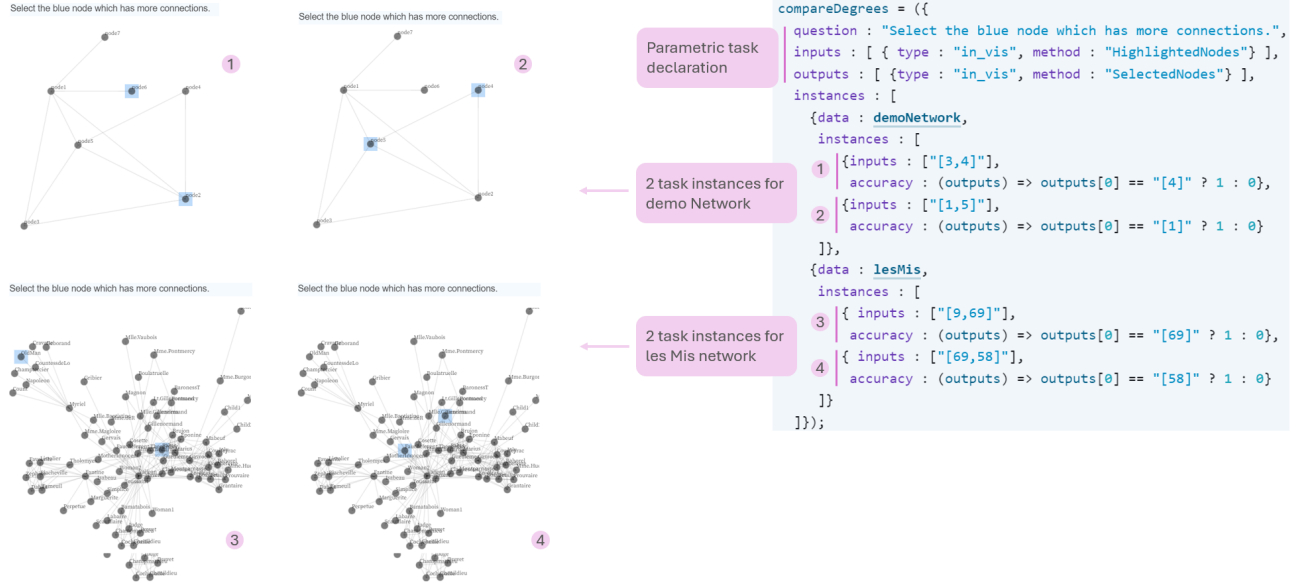
Figure 3: Two visual repeats each for two different datasets (left) emerge from the parameterised task declaration on the right. The task takes two *HighlightedNodes* as input and participants select a node as output (upper right). Two concrete pairs of nodes are picked from each of two networks (*demoNetwork* and *lesMis*) as inputs for four tasks instances (lower-right). The correct answer for each task instance is encoded into its accuracy function.
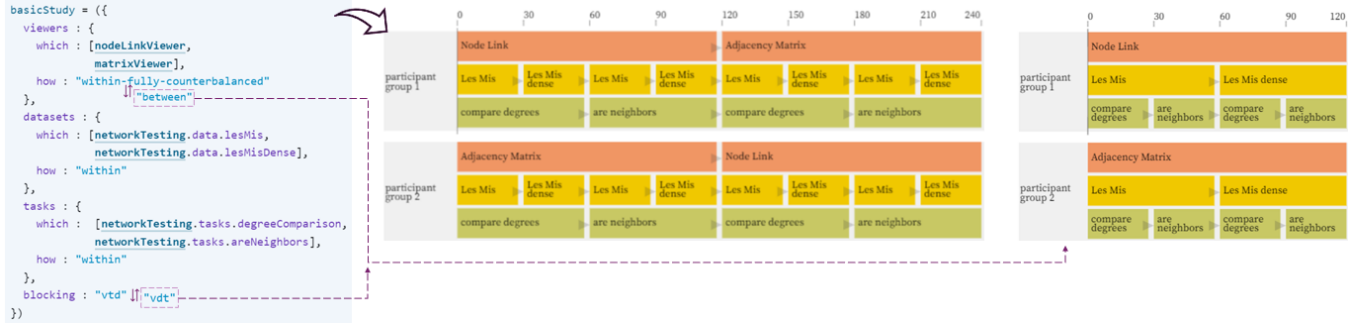

Figure 4: A declarative specification of a 2 viewers × 2 datasets × 2 tasks study, with all factors evaluated within participants and counterbalanced for viewers (left), is translated by VisUnit into a study sequence (middle). VisUnit can then deliver this to participants by sequentially assembling each 'stack' of viewers (orange), datasets (yellow), and tasks (green) into a visual stimulus (see Fig. 5). A small change in the declaration to the viewer evaluation strategy and the study's blocking is seamlessly translated into a new delivery sequence (right).

before moving to the next task. The two tasks included in the declaration are of the form shown in Fig. 3 and include task instances; the two viewers implement the interface methods mentioned in section 3.1 but are otherwise basic Javascript implementations of network visualisations. The datasets contain regular node-link data.

Fig. 4 (centre) shows the study sequence for this declarative design – this is generated automatically by VisUnit from the declarative specification. Since the testing of the viewers is counterbalanced, VisUnit produced distinct sequences for two participant

groups, each group starting with a different visualisation. The benefits of the declarative approach become apparent when small changes made to design specifications translate seamlessly to relatively significant changes in the study delivery. For example, changing the evaluation strategy of viewers to *between participants* and changing the blocking order to *tasks first*, completely rearranges the study sequence as shown in Fig. 4 (right).

Once study sequences produced, VisUnit can assemble the viewers, datasets, and tasks for each factor combination (stacked vertically in Fig.4) into visual stimuli and **deliver the study** in order to participants. VisUnit can collect participants' raw responses, task completion times, and accuracies computed with the help of the tasks' dedicated function.

This basic declarative specification can be extended to capture significantly more complex study designs. Fig. 5 shows a fully functional study constructed around the minimal factorial design discussed above but which was expanded to include training and multiple repeats for each task, introductions of visualisation methods, pre- and post- questionnaires (e.g. visualisation expertise or participant demographics), color-blindness testing, and brief alerts to keep participants informed about upcoming activities and progress. Key principles of its declarative design are shown in Fig.6 and described below:

Customising how a task is delivered: If a task is provided in the format discussed in Sec. 3.1 then a single repeat is chosen from the ones attached to the task. However, tasks can also be provided in an expanded format to include:

- *Number and choice of repeats:* how many repeats of each type of task are to be included and how they are picked from all available repeats (e.g., in order or at random; whether a participant can experience the same repeat in a different task∗data combination or not).
- *Training:* whether tasks ought to be trained, with how many training repeats, with what datasets and when they should be trained (e.g., at the very start of the study or just before a new visualisation is introduced).
- *Estimated task duration:* VisUnit can use this information to estimate the total duration of a study (which is useful when balancing number of conditions and repeats against participant engagement and fatigue) and to inform participants of their progress through the study.
- Mark *gateways tasks* for which participants are required to give a particular answer before they can continue the study. For example, participants need to choose "yes" on the consent page and pass a color-blindness test or else the study will terminate.

Block tasks follow the same format described in Sec. 3.1. However, they are exempt from the factorial design and instead inserted into the study sequence before, after or between factorial blocks. They would be inserted before (`show:"before"`) or after (`show:"after"`) the main (factorial) study (`block:"s"`) in the case for study entry/exit questions such as consent processing, demographic questions, or requests for study-feedback (Fig.6). However, block tasks can also be inserted at different points before, after, or between viewer, dataset, or task blocks (`block:"v"|"d"|"t"`). These would typically be tasks connected to those blocks. Fig. 6 shows the use of a `rate vis` task designed to be introduced after the last showing of each viewer. Again, the benefit of positioning of block tasks relative to the main factorial sequence is that a change to the factorial design leads to an automatic reconfiguring of all activities at no cost to the experimenter.

Parametric and non-parametric tasks. A typical task is parametric – it has *inputs* (data elements) and *outputs* (answers) – and interfaces with data visualisations. However, tasks do not need to have inputs or outputs. **Tasks with no input**s are *non-parametric single instance tasks*. They include tasks that apply to the whole dataset (e.g. "estimate the total number of nodes in the network") or to a visualisation (e.g. the "rate this visualisation"). **Tasks with no outputs** are likely to simply provide information. For example, an `announce demographics` task might simply tells participants that demographic questions will follow. Finally, **tasks with no viewers** rely on task instructions alone and do not need to be used in conjunction with a visualisation or dataset. The "demographic" `askAge` task in Fig.6 is an example of this.

Finally, we note that VisUnit 'visualisations' can be any HTML + Javascript implementations: D3, Vega and other SVG or canvas visualisations; collections of pre-rendered static images displayed as HTML content; simple as well as complex web-pages built around HTML components (e.g., based on React). In the VisUnit model their implementation would load data and revolve primarily around evaluated encodings and interaction designs. However, in practice their implementations can be tailored flexibly to support designs and deliveries that go beyond the default VisUnit model This provides flexibility in the range of studies that can benefit from VisUnit use.

The studies featured in Fig. 4 and 5 can be browsed at https://observablehq.com/@rdjianu/visunit-tutorial.

3.3 Deploying VisUnit studies

VisUnit provides the infrastructure to deliver study sequences (parsed from study specifications) to participants. It assembles stimuli by combining visualisations, dataasets, and tasks, lets participants advance through the study, and collects their responses. The latter include raw participant responses, accuracies computed in accordance with task specifications and those responses, and task timing.

VisUnit does not provide storage for studies or results. VisUnit studies need to be deployed using third-party web hosting services or as part of Observable notebooks as discussed in Sec. 4. Similarly, VisUnit makes new participant responses available as they advance through their studies, but experimenters are expected to provide callback implementations that store these responses into third-party databases.

3.4 Validation

Through replication: We validate VisUnit first by replicating two studies comparing the effectiveness of node-link diagrams and adjacency matrices for displaying network data. Ghoniem *et al.*'s [20] seminal study tested the two visualisations on synthetic networks that varied in size and density (but with overall small network size) and seven types of tasks inspired by Lee *et al.*'s taxonomy [34]. The study was conducted in-person *within* participants. Okoe *et al.* later extended this research to include two real, larger networks, fourteen more diverse tasks – including network clustering and memorability ones — and more interactive visualisations [42, 43]. They conducted their study online with Amazon Mechanical Turk (AMT) participants.

We first sourced the data used in the two experiments. We generated multiple random instances of Ghoniem *et al.*'s 9 categories of graphs (20, 50, 100 nodes × 0.2, 0.4, 0.6 densities) and sourced Okoe
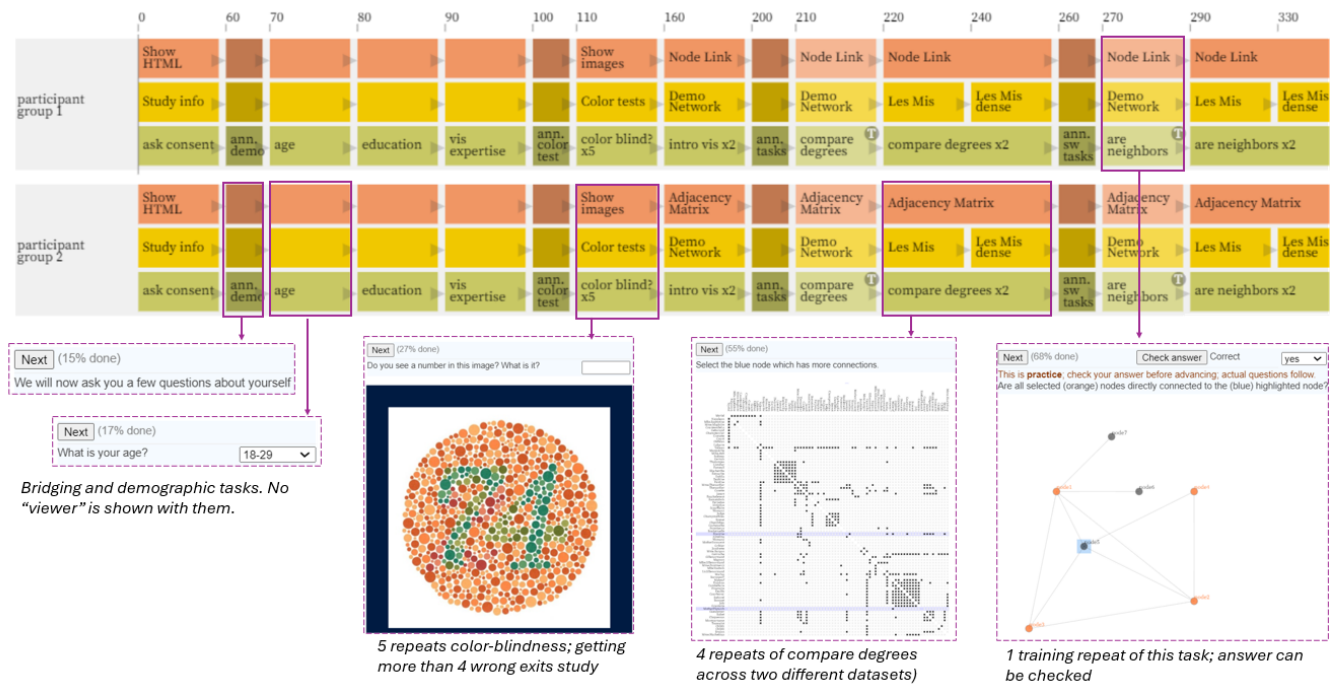
Figure 5: A fully functional study parsed by VisUnit from a declarative specification shown in Fig. 6. It includes, roughly in order (left to right): consent elicitation; demographic questions; testing for color-blindness; trained tasks evaluated over multiple repeats; qualitative questions; and bridging information. Viewers (orange), datasets (yellow) and tasks (green) are combined into visual stimuli exemplified in the bottom half of the image and delivered to participants. VisUnit would collect raw participant responses, task time, and computed accuracy when appropriate.

et al.'s *ingredients* (258 nodes, 1090 edges) and *airports* (332 nodes, 2126 edges) networks. We then defined parameterised VisUnit specifications for all of Ghoniem *et al.*'s 7 tasks and for 10 of Okoe *et al.*'s 14 ones. We skipped their 4 network clustering tasks because, even though straightforward to implement in VisUnit, showing them required more complex visualisations, the development of which was beyond our scope. We kept the two memorability tasks as they introduced complexity in study sequencing: they refer to questions that participants see earlier in a study delivery. We then generated repeats for all task×dataset combinations by choosing concrete data elements as *inputs* for the parameterised tasks. We assembled datasets, tasks and repeats in a network test-suite available here: https://observablehq.com/@rdjianu/network-test-suite. Finally, we implemented node-link and adjacency matrix visualisations with interaction sets to match those used in the replicated studies.

We then assembled the studies using VisUnit's declarative specification. Ghoniem *et al.*'s study was relatively straightforward to set up – *within participant* evaluation of all factors, counterbalanced for visualisations, and with visualisation →network →task blocking. One particularity was in the ordering of the 9 datasets: the 3 sparsest networks were shown before the 6 denser ones but within theses two category the order of graphs was random. VisUnit does not support this by default but can show datasets in a given order. Dataset ordering was thus computed separately and fed into the

study's VisUnit specification. Okoe *et al.*'s setup was a bit more complicated as tasks were divided into three groups; tasks in a group were evaluated within participants but groups themselves between participants. Again, VisUnit does not support task grouping but the setup could be implemented as three different studies, one for each task group, with tasks in each study evaluated within participants. To conclude, both studies could be designed as originally delivered, with minimal extraneous effort. They can be explored at https://observablehq.com/@rdjianu/ghoniem-network-study and https://observablehq.com/@rdjianu/okoe-replica.

By surveying existing studies: To get a more general sense of the range of studies VisUnit supports, we sampled 49 representative visualisation studies and *imagined* whether and how they could be implemented with VisUnit. Fully implementing the studies would not have been feasible. We searched SCOPUS for papers published in Transactions on Visualization and Computer Graphics (TVCG) between 2000 and 2023 with keywords 'user study' and 'evaluation' and 10 citations or more. We read their abstracts and removed papers that did not include quantitative user studies or evaluated visualisations in Virtual or Augmented Reality, on mobile or large displays, or in collaborative contexts. Our rudimentary keyword search missed a few studies known to us and we recognise that there are many other venues where visualisations studies are commonly published. Nevertheless, we judge our sample to be representative of typical visualisation studies.

```
complexDesign = ({
    viewers : {
      which : [nodeLinkViewer2, matrixViewer2],
      how : "within-fully-counterbalanced"},
    datasets : {
      which : [lesMis, lesMisDense],
      how : "within"},
    tasks : {
      which : [
        {
          task : degreeComparison,
          repeats : { number : 2,
                      random :true,
                      unique : true},
          train : { dataset : demoNetwork,
                    with : "v",
                    when : "first showing",
                    repeats : 1,
                    estimated_duration : 10}},
        {
          task : areNeighbors
          /*...options like above*/ }
      ],
      how : "within"
    },
    blocking : "vtd",

    block_tasks : [
      {
        task : askConsent,
        show : "before", block : "s", //show ahead of main (s)tudy
        viewer : HTMLviewer,           //relies on showing
        dataset : consentForms},      //HTML content
      {
        task : ask_age,
        show : "before", block : "s",
        viewer : null},     //show only question
      {
        task : colorBlindTest,
        show : "before",  block : "s",
        viewer : viewer,           //show images
        dataset : dataset,         //of color tests,
        repeats : {number : 5},    //5 of them
        min_accuracy : 0.8},       //80% accuracy needed
      {
        task : rate_vis,          //rate each viewer
        show : "after last", block : "v",
        estimated_duration : 15},
      ]
});
```

Figure 6: Complete declarative specifications let users customise how tasks are delivered (e.g., their number of repeats and whether they are trained) and include tasks that are inserted before, after or between factorial blocks.

We found that 35 studies (71%) we sampled had a direct implementation in VisUnit (A); 5 studies (10%) required minor adjustments in mapping authors' designs and study presentations to the VisUnit structure but could by-and-large be described in VisUnit (B); 9 (18%) were not VisUnit compatible (C). Of such unsupported studies, four followed a staircase design (e.g., [22]); two

of them evaluated stimuli that were not strictly data visualisations – videos and GIFs; in the remaining three, participants' performance was factored dynamically into the choice of stimuli. For example, Boukhelifa *et al.* gave their participants more or fewer experimental blocks depending on their performance[10].

Among studies requiring minor adjustments (B), the most notable issue was a non-straightforward mapping between authors' evaluated factors and the three default VisUnit ones –viewers, datasets, tasks. For example, Borgo *et al.* describe their study on visual embellishments [8] as having two tasks, A and B, with the former having four 'sections', each containing 9 embellished and 9 plain visual stimuli. We mapped 'sections' to VisUnit tasks as they were meant to capture different kinds of activities. We mapped 'stimuli' to nine repeats drawn from a dataset shown with two visualisations. Moreover, the visualisation supplied into VisUnit would need to be engineered to permanently display the author's distractor task B. These adjustments would let VisUnit deliver the study in its original form. Lack of 'study flow' that fits the VisUnit model was another reason. For example, two studies opted for 'semi-factorial' designs: some combinations of factors were included but some not. This can be achieved by letting VisUnit design the study as full factorial, then removing unwanted conditions from resulting study sequences.

4 "Literate", reusable test-suites and studies

4.1 Data+task test suites

The ability to decouple visualisations, datasets and tasks and recombine them flexibly paves the way for *data+tasks test-suites* developed as stand-alone resources to support the design and evaluation of visualisations pursued separately. Such *test-suites* could be constructed to capture tasks specific to certain types of data (e.g., network, multi-dimensional, spatio-temporal, geographic) or to particular domains and applications (e.g. transport analytics, genomics and proteomics, finance and trading, urban decarbonisation).
How? Consider the `compare-degree` task in Fig. 3. Bundling it with the two datasets it references – *demoNetwork* and *lesMis* – provides all information needed to evaluate four variations of this task. VisUnit would be able to set this task up and collect user responses from any (Javascript) network visualisation that lets it highlight nodes (implements `setHighlightedNodes`) and retrieve nodes that users select interactively (implements `getSelectedNodes`).

This task is one listed in Lee *et al.*' taxonomy of network tasks and was used in multiple visualisation studies [6, 19, 25, 43]. We can similarly define other tasks featured in the taxonomy or popular in network evaluation (e.g. `findNode`, `findCommonNeighbour`, `shortestPath`) and instantiate them in our two datasets. We can include additional datasets with different sizes and topological properties. Generally, we can choose data, task presentations, and repeats with evaluation requirements in mind: that they cover how networks are used in practice; reduce confounding factors; and afford meaningful answers. Finally, we can design task presentations such that highlighting nodes and retrieving selected nodes are all the capabilities network visualisations need to set the tasks up.

The result would be a comprehensive *data+task test-suite* to support the evaluation of network visualisations. To be tested, such

visualisations would simply need to implement the two capabilities referenced by the suite's tasks (`setHighlightedNodes` and `getSelectedNodes`). VisUnit would then be able to connect them to the test-suite with minimal overhead. Such test-suites could be developed to capture the particularities and analysis needs of specific types of data, applications, or domains.

Why? Developing such *test-suites* as research artefacts in their own right has several benefits. First, they would reduce the overhead of conducting user studies, help avoid duplication, and supports reuse. This is significant as Quadri *et al.* cites lack of materials as a major roadblock to reuse and replication [44]. Second, it would focus the effort, expertise, and scrutiny needed to develop test-suites that are comprehensive, ecologically valid, meaningful and evolving. While sourcing datasets and tasks to evaluate against typically forms a significant part of a user study [44], it is still just one of several activities involved. As such, the efforts that go into it, as well as the scrutiny it receives, are typically balanced against a study's other activities and contributions. Suboptimal testing materials can thus slip through the research pipeline. Third, they can improve evaluation consistency and support the incremental development of visualisation knowledge. Using a consistent set of tasks and data and presentation to participants would make the results of subsequent user studies more easily relatable.

Finally, *data+task test-suites* complement existing work on task taxonomies. By providing fully fleshed-out task exemplars that can be readily browsed and used in evaluation, they effectively instantiate task taxonomies into 'live' representations that serve similar purposes – design, evaluation – but in more tangible ways.

4.2 Literate test-suites and studies

We advocate that *data+task test-suites* are disseminated as computational notebooks, using "literate" design paradigms [29, 54]. This can bring together all ingredients needed for their scrutiny, interpretation and reuse: testing materials, visual demonstrators, and the reasoning behind their choice and design.

How? *Observable* (observablehq.com) is a notebook environment designed for visualisation and can readily support this. Data and tasks developed within an *Observable* notebook can be easily connected to visualisations and imported into other notebooks for reuse; can be browsed and explored as raw data, programming constructs or visualisations; and are declared in Javascript, currently the *lingua franca* of visualisation. Narratives can be included *in-situ* as markdown text to justify why chosen materials are ecologically valid, sufficiently diverse, and how their setup avoids confounding factors.

Studies as a whole can follow a similar approach. First, *Observable* can be used to store most study ingredients and outputs: materials (either embedded directly or imported from test-suites); their declarative designs; demonstrators of participant delivery; and, once complete, participant responses and their analysis. These can be combined with narratives justifying experimental design; selection of evaluated visual encodings, datasets, and tasks; interpretation of results; and conclusions drawn. The ease of changing declarative studies and observing the effects on participant delivery (Section 3.2), lets researchers explore and showcase different evaluation approaches explicitly (e.g., within vs. between testing,

number of repeats) to support their methodological choices. This is similar to Wood et al.'s branching narratives [54]. Finally, study flows developed and showcased as part of an Observable notebook can be easily embedded in stand-alone web-pages for delivery to participants.

Observable provides native capabilities for reproducing and extending user studies and test-suites through version locking, 'forking' and provenance tracing. Test-suites can be extended over multiple iterations to enrich or tailor their capabilities. For example, a basic test-suite for network visualisation could be forked into extended versions to include attribute-based tasks, dynamic networks, or domain-specific network data and tasks. Similarly, an existing study could be extended by 'forking' it to test different visual encodings or to alter its experimental design. We exemplify these scenarios in Sec. 5.

Why? Overall, the current practice of disseminating user study research and results across many platforms including academic publications (for methodological narratives and result summaries), data and code repositories such as Github or Figshare (for study materials and raw results), and online hosting services (for exemplars and study delivery) makes it difficult to interpret them holistically. Interactive access to all a study's information together and in the open can answer calls for more transparent reporting, dissemination and reuse [21, 30, 44].

4.3 Validation

An exemplar test-suite: To demonstrate the form and utility of *data+task test-suites* we developed a fully functional one for network visualisation: https://observablehq.com/@rdjianu/network-test-suite. It was built primarily around materials used in three user studies on network visualisations, those of Ghoniem *et al.* [20], Okoe *et al.* [43], and Di Giacomo *et al.* [14]. It contains Ghoniem *et al.*'s 9 types of random graphs of different sizes and densities, Okoe *et al.*'s *ingredients* and *airports* networks, and Di Giacomo *et al.*'s *weavers*, *ecoli*, and *dblp* datasets. It also includes 19 different tasks, most of which replicate those evaluated in the three studies above, and repeat instances of these tasks onto the datasets. All tasks include logic necessary to compute the accuracy of participant responses and rely on just two visualisation capabilities to be available: *highlighting* and *selecting* nodes. Finally, the test-suite is augmented with instructions for use and narratives justifying the choice and design of materials. The latter include discussions around the representativeness of datasets and the ecological validity of tasks, their links to taxonomies such as that of Lee *et al.* [34], and how their design minimises confounding factors.

The result is a test-suite that can support the evaluation of network visualisations. It can be readily used to replicate the three studies above with minimal effort as well as to conduct subsequent studies. As long as tested visualisations implement node highlighting and selection, VisUnit can connect them to the test-suite, evaluate them against its data and tasks, and report back accuracy measures. Additionally, the test-suite acts as an instantiation of Lee *et al.*'s network task taxonomy. Users can browse concrete task exemplars across a range of tasks to understand their challenges and imagine visualisation designs to support them better.

An exemplar literate study: We used this test-suite to replicate Okoe *et al.*'s study (described in Sec. 3.4) in 'literate' notebook form: https://observablehq.com/@rdjianu/okoe-replica. We imported datasets and tasks from the test-suite and justified their inclusion into the study (e.g., choice of real datasets over synthetic ones; why networks are representative of real-life networks more broadly; ecological validity of tasks). Similarly, we imported two interactive (Javascript) network visualisations and justified their interaction sets. The notebook was set up to allow the study's materials (datasets, tasks, visualisations) to be explored interactively, helping to contextualise the selection and design narratives provided alongside.

We then combined these materials into a VisUnit study design specification which we linked (using VisUnit utilities) to a schematic view of the study delivery as well as a view of the study as delivered to participants. Through these links, the study's specification (and any changes to it) is seamlessly reflected into the other two views, helping readers observe directly the impact of design choices on study-delivery. Moreover, we showcase the journey towards the study's final presentation as an iteration over alternative designs that were considered (and some abandoned) along the way. As illustrated in Fig. 7, each alternative design is discussed, its specification can be explored, and its participant-facing delivery can be tried in-situ. The ability to directly experience the drawbacks of alternative designs (rather than just imagining them) strengthens the case for the study's ultimate delivery. This mode of alternative design exploration bears similarities to Wood et al.s branching design narratives [54].

Finally, the notebook cells showcasing the study are directly exported (as iFrames) onto separate websites ready for delivery to participants. Participant responses are collected into a database and looped back into the notebook where they can be explored in context with the study design. Thus, the study's materials, designs, delivery, and results are brought together and in the open for inspection and reuse.

5 Use case

We validated individual contributions in Sections 3.4 and 4.3. However, their main benefits stem from how they can impact visualisation evaluation holistically, by supporting study reproduction and extension workflows. We highlight these benefits via **a fictional use case that builds on a sequence of three real studies.** The first two – **(S1)** Ghoniem *et al.*'s study on node-link (NL) diagrams vs. adjacency matrices (AM) and **(S2)** Okoe *et al.*'s subsequent extension of it – were introduced in Sec. 3.4. The third is **(S3)** Di Giacomo *et al.*'s recent evaluation of hybrid network visualisations that blend node-linking, matrices, and chord diagrams [14]. They evaluated six tasks on three real-life networks with online participants recruited from universities and research lists. **We re-imagine this series of studies with VisUnit**:

S1: The authors design a test-suite with 7 tasks and 9 synthetic networks of different sizes and densities. They then create Javascript node-link and adjacency-matrix visualisations and use VisUnit to assemble everything into an Observable "literate" study. They opt for *within participants* evaluation of the visualisations (counterbalanced), datasets and tasks and motivate their design choices (e.g.,
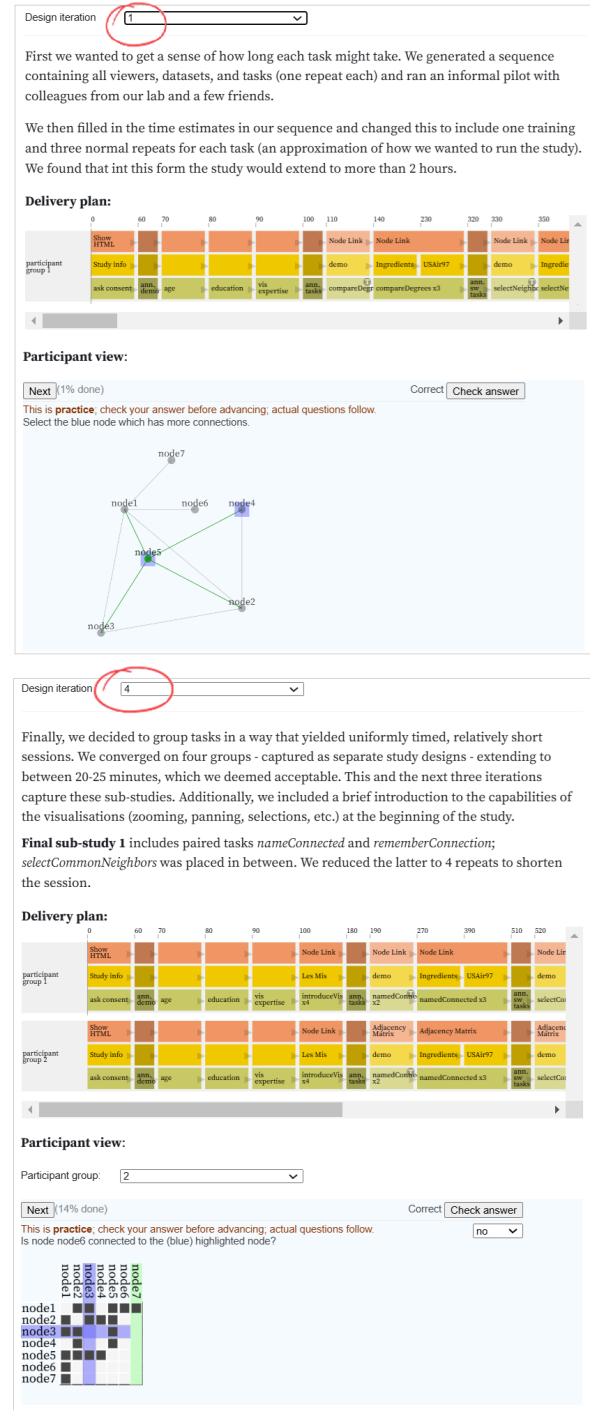

Figure 7: Our literate replica of Okoe et al.'s study captures the evolution of the study's final design as a sequence of iterations. Each iteration is justified and its corresponding delivery is shown and can be experienced interactively. The figure illustrates the first and fourth iteration.

within participants designs are immune to differences in individual participants). Once the study complete, they include de-identified data, result analysis, and interpretation on their study's Observable notebook. They distribute this along with their publication, allowing reviewers to explore their study-design and delivery first hand and to interpret results in context. It also hands future researchers the data and details to extend it.

S2: The authors decide to evaluate more interactive versions of the two network visualisations on real rather than synthetic data and with more tasks. They fork the S1 test-suite and add two datasets to it, one capturing *ingredients* commonly used together in recipes and another of well-connected *airports*. They also define and include 6 new tasks to the test-suite. Along with the data and tasks, they provide in-situ justification of their inclusion (e.g., why real data is preferable to synthetic one; why datasets and tasks are representative). To complete the test-suite expansion, they need to provide repeats for all task × dataset combinations. This means repeats for new S2 tasks on new S2 datasets; but also for old S1 tasks on newly added datasets and for new S2 tasks on the older synthetic datasets. Upgrading the test-suite involves added effort beyond S2 goals but it leads to the creation of a more complete test-suite that now includes both synthetic and real data as well as more tasks.

The design of the study itself is more complex as it involves more tasks and repeats. Moreover, the authors plan to recruit Amazon Mechanical Turk participants who do not engage with long studies. So, delivery needs to be short. Using VisUnit's timing estimate, the authors conclude that a fully *within participant* setup would take about an hour. To half this time, the opt for evaluating the two visualisations *between participants* instead. They also carefully balance the repeats of each task: more for short tasks, fewer for longer tasks. However, studies are still too long. They ultimately decide on a mixture of *between participant* and *within participant* evaluation of tasks.

VisUnit supported this iterative design because changes were quick to make and their impact on study delivery immediately evident. The ability to explore alternative designs gave authors confidence that they converged on a study fit for purpose. The authors describe the evolution towards their final setup in an Observable notebook as a sequence of alternative designs (in both declarative and participant-facing forms) and justifying narratives. Similarly to S1, authors publish de-identified data and result analyses. Together, the information helps reviewers gain a holistic view of the study and its findings.

Finally, S2 authors wish to compare their results to those of S1, to understand the impact of added visualisation interaction or of testing with real rather than synthetic data. However, the significant differences in how the two studies were delivered – S1: *within participants*, in person university students; S2: *between participants*, AMT online users – complicates this. The authors decide to re-run part of the S1 study in its original form but with AMT participants. Setting this up takes just under an hour as the test-suite contains all necessary materials and VisUnit puts everything together. The replication facilitates the desired comparison and new findings.

S3 aims to contribute further with an evaluation of visualisations that blend node linking with matrices. The authors decide to reuse S2's test-suite. The *ingredients* dataset has an appropriate size for their envisioned study, but the *airports* network is too large. As

using a single network would hurt their study's generalisibility, they include two additional datasets, an *ecoli* co-expression network and a *dblp* co-citation one. They decide to use six tasks already available in the test-suite. The inclusion of new datasets means that task repeats need to be provided for all tasks on these datasets. Overall the reuse of tasks and one dataset saved the authors significant time and their effort led to a more complete network evaluation test-suite. Moreover, it means their experimental setup and delivery overlaps significantly that in S2. They can thus relate results to those obtained by S2 and discuss how differences can arise from the slight difference in participant recruitment (students over AMT users).

Summary: When compared to the original research sequence of three independent user studies that each sourced its study materials separately, the imagined use of VisUnit would have resulted in: limited savings for the authors of the initial S1 study; but, a significant reduction in the overhead of conducting S2 and S3; a new, comprehensive network visualisation test-suite to support any future studies; and additional research insights – S2: comparison of in-person vs. AMT delivery; S3: comparison of S3 and S2 results.

6 Discussion

Limitations of the declarative form: VisUnit structures studies around three default factors: *visualisations*, *datasets*, and *tasks*. The benefit is a straightforward mapping from conceptual study design to a model of assembling visual stimuli from granular components – visualisations show datasets and set up tasks – that fits principles of visualisation design and implementation (see Section 3.1). As discussed in Section 3.4, many studies are indeed structured around these three factors. However, some are not (e.g., Borgo *et al.*'s study [8]). In such cases, VisUnit forces authors to map their conceptual factors to its three default ones.

Beyond quantitative studies: While VisUnit was designed with quantitative studies in mind, it can also support qualitative or mixed methods studies. First, tasks can be designed to accept free form answers (without an objectively correct response) and even multiple answers. A task might thus involve a quantifiable response (e.g. a number, a selection) but also qualitative reflection about how the task was performed. Moreover, while VisUnit doesn't itself capture interaction, voice or gaze data, its interfacing mechanisms lets visualisations capture it themselves and return it to VisUnit as components of a task's answer. Second, VisUnit tasks can be designed to capture not just keyhole data-reading tasks, but also more elaborate data analysis scenarios, perhaps with explicit grounding in domain problems. Fig.8 exemplifies the definition and presentation of a relatively complex, domain specific task with multiple outputs - including a selection on the visualisation, a multiple choice selection, qualitative self-reflection, and interactions logged by the visualisation.

Beyond Observable: We advocate that studies be designed and disseminated as visualisation notebooks to support interpretation, scrutiny, and reuse and show how Observable can support this. However, we also distribute VisUnit as a stand-alone library to support studies that need to be created outside the Observable ecosystem. Moreover, we note that Observable notebooks can be exported as stand-alone resources to be stored or pre-registered
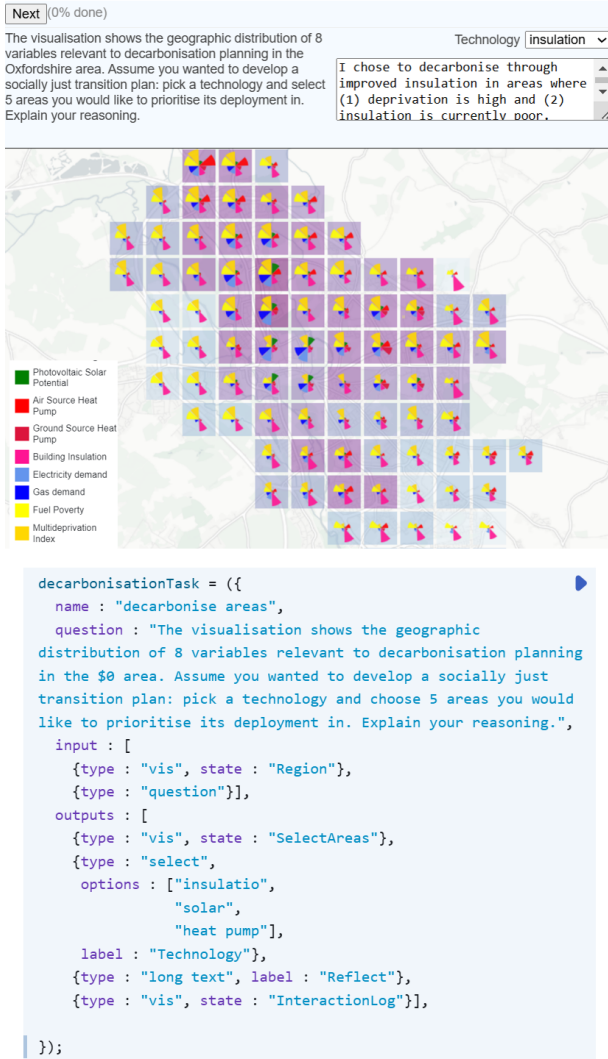
```
Next  (0% done)
The visualisation shows the geographic distribution of 8
variables relevant to decarbonisation planning in the
Oxfordshire area. Assume you wanted to develop a
socially just transition plan: pick a technology and select
5 areas you would like to prioritise its deployment in.
Explain your reasoning.
                                        Technology  insulation ▼
                                 I chose to decarbonise through
                                 improved insulation in areas where
                                 (1) deprivation is high and (2)
                                 insulation is currently poor.
```

Legend:
- Photovoltaic Solar Potential
- Air Source Heat Pump
- Ground Source Heat Pump
- Building Insulation
- Electricity demand
- Gas demand
- Fuel Poverty
- Multideprivation Index

```
decarbonisationTask = ({
  name : "decarbonise areas",
  question : "The visualisation shows the geographic
distribution of 8 variables relevant to decarbonisation planning
in the $0 area. Assume you wanted to develop a socially just
transition plan: pick a technology and choose 5 areas you would
like to prioritise its deployment in. Explain your reasoning.",
  input : [
    {type : "vis", state : "Region"},
    {type : "question"}],
  outputs : [
    {type : "vis", state : "SelectAreas"},
    {type : "select",
     options : ["insulatio",
                "solar",
                "heat pump"],
     label : "Technology"},
    {type : "long text", label : "Reflect"},
    {type : "vis", state : "InteractionLog"}],

});
```

Figure 8: Example of an open ended VisUnit task with multiple outputs to support testing of visualisations for decarbonisation planning.

on other platforms (e.g. Figshare) if needed. Ultimately, while Observable provides a convenient platform to implement 'literate' test-suites and studies, the same principles could be implemented in regular web setups.

Comparison to *reVisIt*: *reVisIt* provides capabilities closest to VisUnit. A series of design principles are also shared: *reVisIt* allows parametric visual stimuli that can serve similar purposes to VisUnit parametric tasks; and VisUnit gateway tasks are similar to *reVisIt* interruption blocks. We have not formally compared VisUnit against reVisIt, but our intuition from validation efforts described in Sec. 3.4, is that *reVisIt*'s declarative language may in fact support a few more designs than VisUnit. On the other hand we point out that this is because *reVisIt*'s specification language is at a lower level of abstraction. For example, its specification and placement of study-blocks is highly recursive and it cannot combine factors as

VisUnit can; adding task repeats to a study is done one at a time and programmatically (compared with a change of parameter in VisUnit); and it doesn't have support for automatic task training. As such, while expressive, we think that reVisIt specifications are longer, and more difficult to design, change and interpret.

Beyond these differences, we feel the main benefit of VisUnit is the explicit decomposition of studies around visualisations, datasets and parametric tasks and the natural delineation of reusable data+task test-suites that this inspires. *reVisIt* provides support for parametric visual stimuli and we think this would be sufficient to implement a form of basic *reVisIt* test-suites. However, *reVisIt* specifications are more monolithic so 'importing' separately defined data and tasks into them is less natural than in VisUnit. Moreover, we see our research as going beyond a system for the development and deployment of user studies: our contributions **(iii)** and **(iv)** are to the visualisation evaluation ecosystem as a whole and they complement work on task taxonomies.

Guiding study design: Through declarative specifications, study design choices with known implications become salient and explicit. This facilitates the provision of study-design guidance and opens the way to *narrative design schemas* suggested by Wood *et al.*[54]. Study designers could be prompted to consider the implications of choosing between a within and between participant setup and provided with information to support their decision. Along with the ability to more easily explore alternative design options, this could lead to studies better fit for purpose.

Default result analyses: Having a formal, declarative study specification paves the way to recommending default result analyses that match the study design declaration. For example, information on whether independent variables were tested between or within participants, along with preliminary tests of data normality, would inform the type of statistical testing needed (e.g. paired t-tests vs. independent t-tests; Wilcoxon rank-sum vs. Wilcoxon signed-rank), and the need for post-hoc testing. While not excluding the possibility of more nuanced, bespoke analysis of raw participant data, default statistical analyses and visualisation of collected data that are aligned with study design would make user studies more accessible to those who lack expertise in numerical methods.

New opportunities: The ability to create and deploy user studies within hours once test-suites assembled, without the need for specialised expertise, can broaden the use of quantitative user evaluations. First, rather than predominantly a way to validate an established visualisation technique or a finished design, lightweight *quantitative evaluations could be integrated within the design process* to help developers choose between alternative designs, similar to how A/B testing is employed in website design. Second, the ability to change design declarations and immediately observe how these are reflected into user study sequences and result analyses can help support the *teaching of user research methods*.

7 Conclusion

We describe a way of specifying the design of visualisation user studies using simple declarative forms and translating them effortlessly into complex participant-facing studies. This encourages the exploration of study design alternatives and reduces the overall overhead of conducting user studies. An important innovation of

our declarative specification is that it lets visualisations, datasets, and tasks underlying studies to be parameterised and defined separately from each other. This paves the way for the development of independent *data+task test suites* for particular data types, applications, or domains and reusing them across studies that evaluate the visual designs meant to support them. They can lower the overhead of running user studies, support reproducibility, and their consistent use can lead to studies with more comparable outcomes. Finally, we advocate for the development of test-suites and studies as visualisation notebooks that bring together and in the open all ingredients necessary for study delivery, scrutiny, and dissemination. We show, with a use case, how embracing these methodologies can lead to more effective, transparent, and cohesive visualisation evaluation.

References

[1] Jae-wook Ahn, Catherine Plaisant, and Ben Shneiderman. 2013. A task taxonomy for network evolution analysis. *IEEE transactions on visualization and computer graphics* 20, 3 (2013), 365–376.

[2] Wolfgang Aigner, Stephan Hoffmann, and Alexander Rind. 2013. Evalbench: A software library for visualization evaluation. In *Computer Graphics Forum*, Vol. 32. Wiley Online Library, 41–50.

[3] Robert Amar, James Eagan, and John Stasko. 2005. Low-level components of analytic activity in information visualization. In *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005*. IEEE, 111–117.

[4] Keith Andrews. 2006. Evaluating information visualisations. In *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*. 1–5.

[5] Natalia Andrienko and Gennady Andrienko. 2006. *Exploratory analysis of spatial and temporal data: a systematic approach*. Springer Science & Business Media.

[6] Lorenzo Angori, Walter Didimo, Fabrizio Montecchiani, Daniele Pagliuca, and Alessandra Tappini. 2020. Hybrid graph visualizations with ChordLink: algorithms, experiments, and applications. *IEEE Transactions on Visualization and Computer Graphics* 28, 2 (2020), 1288–1300.

[7] Roger Beecham, Jason Dykes, Wouter Meulemans, Aidan Slingsby, Cagatay Turkay, and Jo Wood. 2016. Map LineUps: Effects of spatial structure on graphical inference. *IEEE transactions on visualization and computer graphics* 23, 1 (2016), 391–400.

[8] Rita Borgo, Alfie Abdul-Rahman, Farhan Mohamed, Philip W Grant, Irene Reppa, Luciano Floridi, and Min Chen. 2012. An empirical study on using visual embellishments in visualization. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (2012), 2759–2768.

[9] Rita Borgo, Luana Micallef, Benjamin Bach, Fintan McGee, and Bongshin Lee. 2018. Information visualization evaluation using crowdsourcing. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 573–595.

[10] Nadia Boukhelifa, Anastasia Bezerianos, Tobias Isenberg, and Jean-Daniel Fekete. 2012. Evaluating sketchiness as a visual variable for the depiction of qualitative uncertainty. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (2012), 2769–2778.

[11] Matthew Brehmer and Tamara Munzner. 2013. A multi-level typology of abstract visualization tasks. *IEEE transactions on visualization and computer graphics* 19, 12 (2013), 2376–2385.

[12] Jinhan Choi, Changhoon Oh, Yea-Seul Kim, and Nam Wook Kim. 2023. VisLab: Enabling Visualization Designers to Gather Empirically Informed Design Feedback. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–18.

[13] Jinhan Choi, Changhoon Oh, Bongwon Suh, and Nam Wook Kim. 2021. VisLab: Crowdsourcing Visualization Experiments in the Wild. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–7.

[14] Emilio Di Giacomo, Walter Didimo, Giuseppe Liotta, Fabrizio Montecchiani, and Alessandra Tappini. 2023. Comparative study and evaluation of hybrid visualizations of graphs. *IEEE Transactions on Visualization and Computer Graphics* (2023).

[15] Yiren Ding, Jack Wilburn, Hilson Shrestha, Akim Ndlovu, Kiran Gadhave, Carolina Nobre, Alexander Lex, and Lane Harrison. 2023. reVISit: Supporting Scalable Evaluation of Interactive Visualizations. In *2023 IEEE Visualization and Visual Analytics (VIS)*. IEEE, 31–35.

[16] Vijay D'silva, Daniel Kroening, and Georg Weissenbacher. 2008. A survey of automated techniques for formal software verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27, 7 (2008), 1165–1178.

[17] Alexander Eiselmayer, Chat Wacharamanotham, Michel Beaudouin-Lafon, and Wendy E Mackay. 2019. Touchstone2: An interactive environment for exploring trade-offs in hci experiment design. In *Proceedings of the 2019 CHI conference on human factors in computing systems*. 1–11.

[18] Rickard Englund, Sathish Kottravel, and Timo Ropinski. 2016. A crowdsourcing system for integrated and reproducible evaluation in scientific visualization. In *2016 IEEE Pacific Visualization Symposium (PacificVis)*. IEEE, 40–47.

[19] Mohammad Ghoniem, J-D Fekete, and Philippe Castagliola. 2004. A comparison of the readability of graphs using node-link and matrix-based representations. In *IEEE symposium on information visualization*. Ieee, 17–24.

[20] Mohammad Ghoniem, Jean-Daniel Fekete, and Philippe Castagliola. 2005. On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis. *Information visualization* 4, 2 (2005), 114–135.

[21] Steve Haroz. 2018. Open practices in visualization research: Opinion paper. In *2018 IEEE Evaluation and Beyond-Methodological Approaches for Visualization (BELIV)*. IEEE, 46–52.

[22] Lane Harrison, Fumeng Yang, Steven Franconeri, and Remco Chang. 2014. Ranking visualizations of correlation using weber's law. *IEEE transactions on visualization and computer graphics* 20, 12 (2014), 1943–1952.

[23] Jeffrey Heer and Michael Bostock. 2010. Crowdsourcing graphical perception: using mechanical turk to assess visualization design. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 203–212.

[24] Tobias Isenberg, Petra Isenberg, Jian Chen, Michael Sedlmair, and Torsten Möller. 2013. A systematic review on the practice of evaluating visualization. *IEEE Transactions on Visualization and Computer Graphics* 19, 12 (2013), 2818–2827.

[25] Radu Jianu, Adrian Rusu, Yifan Hu, and Douglas Taggart. 2014. How to display group information on node-link diagrams: An evaluation. *IEEE Transactions on Visualization and Computer Graphics* 20, 11 (2014), 1530–1541.

[26] Natalie Kerracher, Jessie Kennedy, and Kevin Chalmers. 2015. A task taxonomy for temporal graph visualisation. *IEEE transactions on visualization and computer graphics* 21, 10 (2015), 1160–1172.

[27] Vladimir Khorikov. 2020. *Unit Testing Principles, Practices, and Patterns*. Simon and Schuster.

[28] Gordon Kindlmann and Carlos Scheidegger. 2014. An algebraic process for visualization design. *IEEE transactions on visualization and computer graphics* 20, 12 (2014), 2181–2190.

[29] Robert Kosara. 2023. Notebooks for data analysis and visualization: Moving beyond the data. *IEEE Computer Graphics and Applications* 43, 1 (2023), 91–96.

[30] Robert Kosara and Steve Haroz. 2018. Skipping the replication crisis in visualization: Threats to study validity and how to address them: Position paper. In *2018 IEEE Evaluation and Beyond-Methodological Approaches for Visualization (BELIV)*. IEEE, 102–107.

[31] Robert Kosara, Christopher G Healey, Victoria Interrante, David H Laidlaw, and Colin Ware. 2003. Thoughts on user studies: Why, how, and when. *IEEE Computer Graphics and Applications* 23, 4 (2003), 20–25.

[32] Robert Kosara and Caroline Ziemkiewicz. 2010. Do mechanical turks dream of square pie charts?. In *Proceedings of the 3rd BELIV'10 Workshop: Beyond time and errors: Novel evaluation methods for information visualization*. 63–70.

[33] Heidi Lam, Enrico Bertini, Petra Isenberg, Catherine Plaisant, and Sheelagh Carpendale. 2011. Empirical studies in information visualization: Seven scenarios. *IEEE transactions on visualization and computer graphics* 18, 9 (2011), 1520–1536.

[34] Bongshin Lee, Catherine Plaisant, Cynthia Sims Parr, Jean-Daniel Fekete, and Nathalie Henry. 2006. Task taxonomy for graph visualization. In *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*. 1–5.

[35] Laura E Matzen, Michael J Haass, Kristin M Divis, Zhiyuan Wang, and Andrew T Wilson. 2017. Data visualization saliency model: A tool for evaluating abstract data visualizations. *IEEE transactions on visualization and computer graphics* 24, 1 (2017), 563–573.

[36] Andrew McNutt and Gordon Kindlmann. 2018. Linting for visualization: Towards a practical automated visualization guidance system. In *VisGuides: 2nd Workshop on the Creation, Curation, Critique and Conditioning of Principles and Guidelines in Visualization*, Vol. 1. 2.

[37] Monique Meuschke, Noeska N Smit, Nils Lichtenberg, Bernhard Preim, and Kai Lawonn. 2019. EvalViz–Surface visualization evaluation wizard for depth and shape perception tasks. *Computers & Graphics* 82 (2019), 250–263.

[38] Silvia Miksch and Wolfgang Aigner. 2014. A matter of time: Applying a data–users–tasks design triangle to visual analytics of time-oriented data. *Computers & Graphics* 38 (2014), 286–290.

[39] Tamara Munzner. 2009. A nested model for visualization design and validation. *IEEE transactions on visualization and computer graphics* 15, 6 (2009), 921–928.

[40] Tamara Munzner. 2014. *Visualization analysis and design*. CRC press.

[41] Mershack Okoe and Radu Jianu. 2015. Graphunit: Evaluating interactive graph visualizations using crowdsourcing. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 451–460.

[42] Mershack Okoe, Radu Jianu, and Stephen Kobourov. 2018. Node-link or adjacency matrices: Old question, new insights. *IEEE transactions on visualization and computer graphics* 25, 10 (2018), 2940–2952.

[43] Mershack Okoe, Radu Jianu, and Stephen Kobourov. 2018. Revisited experimental comparison of node-link and matrix representations. In *Graph Drawing and Network Visualization: 25th International Symposium, GD 2017, Boston, MA, USA, September 25-27, 2017, Revised Selected Papers 25*. Springer, 287–302.

[44] Ghulam Jilani Quadri and Paul Rosen. 2021. A survey of perception-based visualization studies by task. *IEEE transactions on visualization and computer graphics* 28, 12 (2021), 5026–5048.

[45] Robert E Roth. 2013. An empirically-derived taxonomy of interaction primitives for interactive cartography and geovisualization. *IEEE transactions on visualization and computer graphics* 19, 12 (2013), 2356–2365.

[46] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. 2016. Vega-lite: A grammar of interactive graphics. *IEEE transactions on visualization and computer graphics* 23, 1 (2016), 341–350.

[47] Georgiy Shurkhovetskyy, Natalia Andrienko, Gennady Andrienko, and Georg Fuchs. 2018. Data abstraction for visualizing large time series. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 125–144.

[48] Melanie Tory, David Sprague, Fuqu Wu, Wing Yan So, and Tamara Munzner. 2007. Spatialization design: Comparing points and landscapes. *IEEE Transactions on Visualization and Computer Graphics* 13, 6 (2007), 1262–1269.

[49] Terece L Turton, Anne S Berres, David H Rogers, and James P Ahrens. 2017. ETK: An Evaluation Toolkit for Visualization User Studies.. In *EuroVis (Short Papers)*. 43–47.

[50] Jenny Vuong, Sandeep Kaur, Julian Heinrich, Bosco K Ho, Christopher J Hammang, Benedetta F Baldi, and Seán I O'Donoghue. 2018. Versus—A tool for

[51] evaluating visualizations and image quality using a 2AFC methodology. *Visual Informatics* 2, 4 (2018), 225–234.

[51] Emily Wall, Meeshu Agnihotri, Laura Matzen, Kristin Divis, Michael Haass, Alex Endert, and John Stasko. 2018. A heuristic approach to value-driven evaluation of visualizations. *IEEE transactions on visualization and computer graphics* 25, 1 (2018), 491–500.

[52] Stephen Wehrend and Clayton Lewis. 1990. A problem-oriented classification of visualization techniques. In *Proceedings of the First IEEE Conference on Visualization: Visualization90*. IEEE, 139–143.

[53] Leland Wilkinson. 2012. *The grammar of graphics.* Springer.

[54] Jo Wood, Alexander Kachkaev, and Jason Dykes. 2018. Design exposition with literate visualization. *IEEE transactions on visualization and computer graphics* 25, 1 (2018), 759–768.

[55] Aoyu Wu, Yun Wang, Xinhuan Shu, Dominik Moritz, Weiwei Cui, Haidong Zhang, Dongmei Zhang, and Huamin Qu. 2021. Ai4vis: Survey on artificial intelligence approaches for data visualization. *IEEE Transactions on Visualization and Computer Graphics* 28, 12 (2021), 5049–5070.

[56] Michelle X Zhou and Steven K Feiner. 1998. Visual task characterization for automated visual discourse synthesis. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 392–399.